AD-A097 608 SRI INTERNATIONAL MENLO PARK CA F/G 12/2
RESEARCH ON PARALLELISM IN PROBLEM-SOLVING SYSTEMS.(U)
SEP 80 D E WILKINS F49620-79-C-0188
UNCLASSIFIED AFOSR-TR-80-1150 NL

1 OF 1
AD
A097608

END
DATE
FILMED
5 81
DTIC

SRI International

RESEARCH ON PARALLELISM IN PROBLEM-SOLVING SYSTEMS

September 31, 1980

1st Annual Technical Report

SRI Project 8871

By: David E. Wilkins
Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Air Force Office of Scientific Research
Building 410
Bolling Air Force Base
Washington, D. C.

Attention: Captain William Price
Contract No. F49620-79-C-0188

333 Ravenswood Ave. • Menlo Park, California 94025
(415) 326-6200 • Cable: SRI INTL MPK • TWX: 910-373-1246

I. INTRODUCTION

This report summarizes the first year of research on a system for automatically generating hierarchical plans containing parallel (concurrent) actions. This is a general planning and problem-solving system that is not tied to a particular domain. Results of this research might eventually be used in the development of systems for automatically generating plans to coordinate the activities of military personnel engaged in a common mission. Other domains of application could include logistical planning, planning to use many computers on a network concurrently, and planning to prepare and disseminate a report or other information to a large group of people.

General-purpose problem-solving systems elicited wide interest in the early and mid-1970s, and Sacerdoti's NOAH was one of the most important general planners produced. It remains a landmark because there was little activity in this area during the late 1970s. The past year of research on this project has resulted in the following accomplishments:

1. Analysis of the shortcomings of NOAH with ideas for improving it.
2. Design of a general planning system that advances the state of the art while eliminating these shortcomings.
3. Implementation started of the designed system in a computer program.

Some of the results of the new design appeared in a technical publication which is reproduced in the Appendix. The project is now in its second year, during which time all parts of the system will be designed in detail and the implementation will be made more complete. This report summarizes research performed during the past year.

II. ANALYSIS OF PROBLEM

Our work began by studying the NOAH system and attempting to improve upon it. Three specific improvements were made. A procedure was written for printing procedural nets graphically to aid analysis. Upon finding a conflict in two parallel branches of a plan NOAH could solve the problem only by doing one branch before the other. SIGNAL and WAIT nodes were added this year to allow synchronization at various points along the parallel branches. A new critic was also added to NOAH. Goals in NOAH's plans that at first appear to have been already achieved, may no longer appear so after certain actions are taken. The new critic checks for such goals.

During this study of NOAH, we outlined a number of major weaknesses and developed ideas for a planning system without these weaknesses. NOAH made some fundamental assumptions that prevented the application of our solutions, so we decided to create a new planning system rather than continue to improve upon NOAH. The new system will advance the state of the art in general planning systems in the following seven areas:

1. There will be much more domain knowledge in the system, including a type hierarchy with automatic inheritance of properties.
2. The system will be able to investigate many alternative plans simultaneously.
3. A more general, flexible, and usable language for describing actions will be developed.
4. The system will be able to describe its own planning actions in its language for describing actions, providing the groundwork for a metaplanning ability in which the system can reason about its own abilities.
5. The system will know about resources, allowing for their easy specification and automatic allocation. This allows for a faster and better analysis of the interactions among parallel branches of a plan.
6. The system will be able to partially describe variables without binding them to specific objects. This allows great flexibility in the description of parallel actions.
7. The system will be able to replan after certain unexpected occurrences during execution without redoing the whole plan.

III. DESIGN AND IMPLEMENTATION OF THE NEW SYSTEM

During the past year, the overall design of the new system has been done at a high level, and a detailed design for some of the areas mentioned in the last section has been accomplished. The following areas have been designed in detail:

1. The type hierarchy, including methods for creating and accessing it.
2. The ability to create, store, and access many alternative plans (easy ways to switch focus from one plan to another still need to be designed).
3. A general, flexible and easy-to-use language for describing actions.
4. Methods for specifying resources (neither allocating resources nor finding conflicts on parallel branches has been done yet).
5. Methods for partially describing variables by posting constraints.

During the past year we began to implement the new system in INTERLISP. Most of the areas mentioned above as being designed in detail have been coded. However, the code is just bits and pieces; there is, as yet, no system than can plan to accomplish a goal from start to finish.

IV. FUTURE WORK

In the upcoming year we intend to finish the detailed design of all seven areas of the new planning system, with the possible exception of the metaplanning abilities. The ability to allocate resources and solve conflicts among parallel branches of a plan will be a majcr part of this effort. We intend to complete the implementation of nearly all the above ideas to produce a planning system that will offer these capabilities. We intend to test the system in sample domains and to consider possible military applications.

V. PUBLICATIONS AND CONFERENCES

This project supported in part the attendance of David Wilkins and Ann Robinson at the Canadian Society for Computational Studies of Intelligence Conference at the University of Victoria, 14-16 May 1980, and at the American Association for Artificial Intelligence Conference at Stanford University, 19-21 August 1980. Their paper in the proceedings of the latter conference reports research done on this project and is included in the Appendix of this report.

REPRESENTING KNOWLEDGE IN AN INTERACTIVE PLANNER

Ann E. Robinson and David E. Wilkins
Artificial Intelligence Center
SRI International
Menlo Park, California 94025

ABSTRACT

This note discusses the representation for actions and plans being developed as part of the current planning research at SRI. Described is a method for uniformly representing actions that can take place both in the domain and during planning. The representation accommodates descriptions of abstract (hypothetical) objects.

I. INTRODUCTION

A principal goal of current planning and plan-execution research at SRI is development of a planning system that interacts with a person, allowing that person to: (1) explore alternative plans for performing some activity, (2) monitor the execution of a plan that has been produced, and (3) modify the plan as needed during execution.* Described here is the knowledge representation being developed.

Our research builds directly on previous planning research and on research in representing the domain knowledge necessary for participating in natural-language dialogs about tasks. In particular, some of our representation ideas are based on the process model formalism described in [2] and [3]. The basic approach to planning is to work within the hierarchical planning paradigm, representing plans in procedural networks, as has been done in NOAH [4] and other systems.

----

* The research reported here is supported by Air Force Office of Scientific research Contract N00014-80-C-0300 and by Office of Naval Research Contract F49620-79-C-0188.

Unlike its predecessors, our new system is being designed to allow interaction with users throughout the planning and plan-execution processes. The user will be able to watch and, when desired, guide and/or control the planning process. During execution of a plan, some person or computer system monitoring the execution will be able to specify what actions have been performed and what changes have occurred in the world being modeled. On the basis of this, the plan can be interactively updated to accommodate unanticipated occurrences. Planning and plan-execution can be intermingled by producing a plan for part of an activity and then executing some or all of that plan before working out remaining details.

We are extending planning research in several major directions. One of the key directions, the one discussed here, is a method for representing actions that can take place both in the domain and during planning. Action descriptions (often referred to as operators), procedural networks, and knowledge about domain objects and their interrelationships are represented in the same formalism -- a hierarchy of nodes with attributes. This uniform representation provides the ability to encode partial descriptions of unspecified objects as well as objects in the domain model. Thus, operator descriptions referring to abstract (unbound) objects can be represented in the same formalism as procedural network nodes referring to specific objects in the domain model. (Partial descriptions of unspecified objects will be described here as constraints on the possible values of a variable representing the object.)

Objects involved in an action often can be characterized as resources that are to be used during a particular action and then released, e.g., a saw used during a cutting action. Since this is a common phenomenon and since it is often difficult or awkward to keep track of resources in current planning systems, we have included in the formalism a means of specifying the objects that are resources for an action. Declaration of a resource implicitly specifies preconditions on the availability of the resource, and processes in the planning system automatically satisfy these preconditions as they allocate and deallocate resources.

Operators can be encoded at several levels of abstraction. Each one contains information for planning at the next level of detail. We have already encoded many domain operators for a construction task; planning operators will be encoded shortly. The domain operators provide the planning system with information about producing a plan in the domain. The planning operators provide the planning system with information so it can reason about its own planning process (meta-planning). They also provide a major part of the interface between the planning system and the user, who will be able to direct the planning process via the planning operators.

The uniformity of representation for domain knowledge, specific plans of action, and all operators will facilitate both the user's ability to interact with and control the planning system, and the system's ability to incorporate (learn) new operators from plans it has already produced. We will describe the representation in more detail below.

II. THE FORMALISM

The formalism for representing knowledge about actions, plans, and domain objects consists of typed nodes linked in a hierarchy. Each node can have attributes associated with it. There are four node types for representing objects: CLASS, INSTANCE, INDEFINITE, and DESCRIPTION. These will not be discussed in more detail here since they are similar to those occurring in representation formalisms such as KRL, FRL, and UNITS [5].

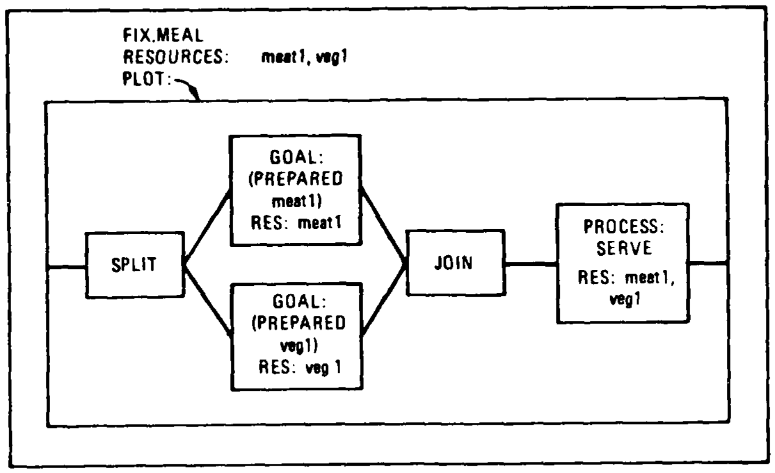
The node types for representing actions can be grouped into four categories:

OPERATOR, for encoding operators;
PNET, for representing specific actions (nodes in the procedural network);
PLOT, for describing how to expand a given OPERATOR, i.e., a description of an action in greater detail;
PNET.ACTION, for encoding plan steps (procedural network nodes) that have been 'executed' and thus represent actions assumed to have occurred in the world being modeled.

Nodes can have lists of attributes and can be connected into a hierarchy through CLASS and SUBCLASS links. Attributes of nodes for representing actions include the resources and arguments of the action (i.e., the objects that participate in the action), the action's goal, the action's effects on the domain when it is performed, and the action's preconditions. OPERATOR nodes have a plot attribute which specifies PLOT nodes for carrying out the operator.

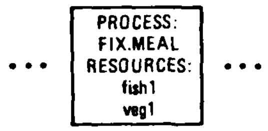
The PLOT of an operator can be described not only in terms of GOALs to be achieved, but also in terms of PROCESSes to be invoked. (Previous systems would represent a PROCESS as a goal with only a single choice for an action to perform.) The ability to describe operators in terms of both GOALs and PROCESSes will help simplify encoding of operators and will allow the planning system to reason about alternative action sequences more efficiently.

OPERATOR:

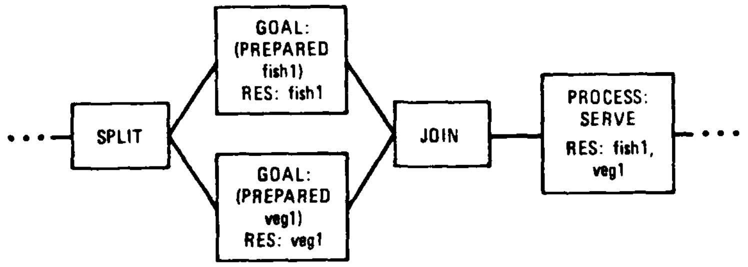


FIX.MEAL
RESOURCES: meat1, veg1
PLOT:
SPLIT
GOAL: (PREPARED meat1) RES: meat1
GOAL: (PREPARED veg1) RES: veg 1
JOIN
PROCESS: SERVE RES: meat1, veg1

A PNET node which represents using FIX.MEAL in a plan:



Expansion of this PNET node at the next level using FIX.MEAL:



SPLIT
GOAL: (PREPARED fish1) RES: fish1
GOAL: (PREPARED veg1) RES: veg1
JOIN
PROCESS: SERVE RES: fish1, veg1

Figure 1

Figure 1 shows a sample operator and a PNET it might produce. The figure illustrates the uniformity across different types of nodes in our formalism. The nodes are expressed in the same formalism, and, for the most part, have the same attributes (e.g., resources, shared-resources, arguments, preconditions, purpose) with similar values for these attributes. "Similar values" means that the values refer to the same types of objects -- often the value of an attribute for some node will be more constrained than the value of the same attribute in a corresponding node of a different category. The next two paragraphs illustrate this in detail, after which we describe two instances where the uniformity of the representation is advantageous.

Attributes in OPERATOR and PLOT nodes generally refer to variables rather than specific objects, since these are uninstantiated operators that may be instantiated into PNET nodes in different ways during planning. For example, in Figure 1 resource variables meat1 and veg1 in the operator FIX.MEAL refer to objects of the meat and vegetable class, respectively. In the expansion of FIX.MEAL, meat1 has been constrained to be a fish (denoted by calling it "fish1") since it was so constrained in the node being expanded.

In our formalism, such variables are described by INDEFINITE nodes with constraints on their possible values. For PNET nodes, attributes frequently refer both to variables (which will often be more constrained in this case) and to completely specified objects. For PNET.ACTION nodes, attributes generally refer to specific objects in the domain model. The system's ability to use INDEFINITE nodes to partially describe objects is important for representing objects with varying degrees of abstractness in the same formalism. Few previous planning systems have used this approach (e.g., NOAH cannot partially describe objects and has different formalisms for describing operators and procedural nets). Stefik's system [5] does allow abstract descriptions and constraints on partially described arguments, but arguments are required to be fully instantiated before the constraints can be evaluated. (See also Hayes-Roth et al. [1].)

The uniformity of representation between PLOT and PNET nodes permits the description of operators as what amounts to generalized fragments of procedural network. This turns problem solving into a process of incremental instantiation. During planning, PNET nodes are incrementally expanded to a greater level of detail by selecting an appropriate operator, determining which of its variables match those in the node being expanded, creating new variable records for those variables not matched, adding any new constraints to these variables, and following the operator's plot description to create new procedural network nodes. The uniformity of representation facilitates this production of PNET nodes from PLOT nodes.

Once a plan has been successfully constructed, it may be desirable to save it for subsequent planning activities, incorporating it into the system as a new operator. We expect to develop algorithms for doing this, i.e., producing an operator (with its associated PLOT nodes) from PNET fragments. For each control node and each action-oriented node in a procedural network, a corresponding PLOT node can be easily created for the operator because of the uniformity of representation. The major task remaining in producing an operator would be generalizing the constraints on values for variables in the procedural network nodes into looser constraints in the new operator.

An additional uniformity between descriptions of specific actions and operators facilitates the matching of an operator to the node it is to expand. Thus PROCESS and GOAL nodes in a procedural network or plot will have attributes similar to those of the OPERATOR node which represents a more detailed description of their corresponding action. The similarities of representation of all action-oriented nodes facilitates interaction with the user who can talk in the same way about operators, steps in operator plots, and nodes in the procedural network. Similarly, description of actions is facilitated by this uniformity.

Organizing the representation as nodes with attributes is, of course, not new and is not essential. The representation could also be expressed in a formal logic (a translation to logic would be fairly straightforward). We have chosen to represent things as nodes with attributes because this blends well with our plans for interaction with the user.

III. PARTIAL DESCRIPTION USING CONSTRAINTS

Stefik's system [5], one of the few existing planning systems with the ability to construct partial descriptions of an object without identifying the object, contains a constraint-posting mechanism that allows partial descriptions similar to those described above. Our system also provides for partial description using constraints, and extends Stefik's approach in two ways.

Unlike Stefik's system, our system permits evaluation of constraints on partially described objects. Both CLASSes and INSTANCEs can have constraints. For example, a set can be created which can be constrained to be only bolts, then to be longer than one inch and shorter than two inches, and then to have hex heads. Our system also provides for partial descriptions that vary with the context, thus permitting consideration of alternative plans simultaneously. A context mechanism has been developed to allow for alternative constraints on variables relative to different plan steps. The constraints on a variable's value as well as the binding of a variable to a particular instance (possibly determined during the solution of a general constraint-satisfaction problem) can only be retrieved relative to a particular context. This permits the user to easily shift focus back and forth between alternatives. Hayes-Roth et al. [1] describe the use of a blackboard model for allowing shifting of focus between alternatives. Such focus shifting can not be done in systems using a backtracking algorithm where descriptions built up during expansion of one alternative are removed during the backtracking process before another alternative is investigated. Most other planning systems either do not allow alternatives (e.g., NOAH[4]), or use a backtracking algorithm (e.g., Stefik [5], Tate [6]).

IV. CONCLUSION

We have described some properties of the knowledge representation developed for our new planning system. Most of the planner is still under development (e.g., critics, reasoning about resources, and search control have yet to be implemented). The central idea discussed here is the uniform representation of the domain operators, planning operators, procedural networks, and knowledge about domain objects. Ways to exploit this uniformity are pointed to. These include a rich interaction with the user, meta-planning, and having the system learn new operators from plans it has constructed.

REFERENCES

1. Hayes-Roth, B., F. Hayes-Roth, S. Rosenschein, and S. Cammarata, "Modeling Planning as an Incremental, Opportunistic Process" In Proc. IJCAI-79. Tokyo, Japan, August, 1979. pp. 375-383.

2. Hendrix, G., "Encoding Knowledge in Partitioned Networks." In Associative Networks-The Representation and Use of Knowledge in Computers, N.V. Findler, ed., Academic Press, New York, New York, 1979.

3. Robinson, A.E., D. Appelt, B. Grosz, G. Hendrix, and J. Robinson, "Interpreting Natural-Language Utterances in Dialogs About Tasks", Technical Note 210, SRI International, Menlo Park, California. March, 1980.

4. Sacerdoti, E., A Structure for Plans and Behavior. Elsevier North-Holland, New York, 1977.

5. Stefik, M., Planning With Constraints. Report No. STAN-CS-80-784, Computer Science Department, Stanford University, Ph.D Dissertation. 1980.

6. Tate, A., "Generating Project Networks", In Proc. IJCAI 5. Cambridge, Massachusetts, August, 1977.